AN EVALUATION AND COMPARISON OF SEVERAL SINGLE VARIABLE SEARCH METHODS

Daniel Brian Wick

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

AN EVALUATION AND COMPARISON OF
SEVERAL SINGLE VARIABLE SEARCH METHODS

by

Daniel Brian Wick

June 1976

Thesis Advisor: J.K. Hartman

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER

2. GOVT ACCESSION NO.

3. RECIPIENT'S CATALOG NUMBER

4. TITLE (and Subtitle)
An Evaluation and Comparison of Several Single Variable Search Methods

5. TYPE OF REPORT & PERIOD COVERED
Master's Thesis; June 1976

6. PERFORMING ORG. REPORT NUMBER

7. AUTHOR(s)
Daniel Brian Wick

8. CONTRACT OR GRANT NUMBER(s)

9. PERFORMING ORGANIZATION NAME AND ADDRESS
Naval Postgraduate School
Monterey, California 93940

10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS

11. CONTROLLING OFFICE NAME AND ADDRESS
Naval Postgraduate School
Monterey, California 93940

12. REPORT DATE
June 1976

13. NUMBER OF PAGES
76

14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)

15. SECURITY CLASS. (of this report)
Unclassified

15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)
Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)
Single Variable Search Method
Golden Section
cubic interpolation
quadratic interpolation
Nonlinear programming

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This study compares three single variable search methods — Golden Section, cubic interpolation and quadratic interpolation. The SUMT nonlinear program was used for the comparison. The OPT subroutine which performs the single variable search in SUMT currently uses the Golden Section method. Two different OPT subroutines were written which implemented cubic interpolation and quadratic interpolation. Seven test

(20. ABSTRACT Continued)

problems which contained 9-100 variables and 2-20 constraints were used. The comparison was made on computation time per single variable search for the three methods and the number of function evaluations per single variable search for the Golden Section and quadratic interpolation methods.

A single variable search by Lasdon, Fox and Ratner and one by Fletcher and McCann were also discussed.

The results showed that the quadratic interpolation was slightly faster than the other two methods and required fewer function evaluations per single variable search than the Golden Section method. Time per single variable search was approximately the same for the cubic interpolation and Golden Section methods. Cubic interpolation required fewer points to be evaluated than the other two methods but the need for gradient evaluations proved to be costly in terms of computation time per single variable search.

An Evaluation and Comparison of Several Single Variable Search Methods

by

Daniel Brian Wick
Ensign, United States Navy
B.S., United States Naval Academy, 1975

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN OPERATIONS RESEARCH

from the

NAVAL POSTGRADUATE SCHOOL
June 1976

ABSTRACT

This study compares three single variable search methods – Golden Section, cubic interpolation and quadratic interpolation. The SUMT nonlinear program was used for the comparison. The OPT subroutine which performs the single variable search in SUMT currently uses the Golden Section method. Two different OPT subroutines were written which implemented cubic interpolation and quadratic interpolation. Seven test problems which contained 9-100 variables and 2-20 constraints were used. The comparison was made on computation time per single variable search for the three methods and the number of function evaluations per single variable search for the Golden Section and quadratic interpolation methods.

A single variable search by Lasdon, Fox and Ratner and one by Fletcher and McCann were also discussed.

The results showed that the quadratic interpolation was slightly faster than the other two methods and required fewer function evaluations per single variable search than the Golden Section method. Time per single variable search was approximately the same for the cubic interpolation and Golden Section methods. Cubic interpolation required fewer points to be evaluated than the other two methods but the need for gradient evaluations proved to be costly in terms of computation time per single variable search.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

I. INTRODUCTION

The concept of optimization has grown to play a major role in the analysis of many complex decision and allocation problems. The quality of the analysis not only depends upon the skill and good judgement used in interpreting and modeling the problem but also upon the reliability and efficiency of the vehicle used for finding a solution to the problem. The solving of nonlinear programming problems has seen substantial development during the past twenty years and, no doubt, will increase in acceptance and popularity in the future.

Within the Department of Defense several problems of significant importance, such as weapons targeting and allocation, aircraft and power plant design and manpower planning, to name a few, have arisen which are nonlinear in nature. Some examples of industrial nonlinear programming problems include oil refining, curve fitting, inventory and logistics.

Many methods exist for solving nonlinear programming problems. Some of these methods, called penalty function methods, are based on transforming a constrained minimization problem into a sequence of unconstrained minimization problems whose solutions approach the solution of the original constrained problem.

In the penalty function method, as well as in most other nonlinear programming algorithms, a one dimensional

minimization search is used repeatedly in the solution process. In this paper several different one dimensional minimization search methods are compared in the context of penalty function algorithms.

The different one dimensional minimization search methods (also called single variable searches) are compared by the amount of computation time and by the number of function, gradient and Hessian evaluations of the objective function and constraints needed to find a solution to the original constrained optimization problem.

Section II of this paper explains the formulation of the penalty function method and where the single variable search is used in this method. The nonlinear computer program SUMT is used in comparing the different single variable search methods and is also explained.

Section III formulates the different single variable search methods — Golden Section, cubic interpolation and quadratic interpolation — and also discusses several other methods which were not programmed because of their complexity.

Results and comparisons of the different methods are stated in Section IV with conclusions and suggestions for further study in Section V. The Appendix contains the computer codes for the three methods that were programmed.

II. THE PENALTY FUNCTION METHOD

A. THE OPTIMIZATION PROBLEM

The single variable search is a major part of finding an optimal solution to a constrained optimization problem by the penalty function method. Dr. W.C. Mylander, an author of the SUMT computer code, estimated that forty percent of the computation time required to find an optimization problem solution using SUMT involved single variable searches.

The constrained optimization problem to be solved is of the form:

$$\begin{aligned} &\text{minimize:} && f(x) \\ &\text{subject to:} && g_j(x) \geq 0 \qquad j = 1,\ldots,m \\ &&& h_j(x) = 0 \qquad j = m+1,\ldots,m+p \ , \end{aligned}$$

where x is an n-dimensional vector, the functions f, g and h are continuous and may be either linear or nonlinear, and the set of values satisfying the inequality constraints has a non-empty interior, i.e. $\{x: g_j(x) > 0,\ j=1,\ldots,m\}$.

B. THE SUMT COMPUTER PROGRAM

The computer program used to compare the different single variable search methods was SUMT - Version 4 (Sequential Unconstrained Minimization Technique for Nonlinear Programming) coded in FORTRAN IV by W.C. Mylander, R.L. Holmes and G.P. McCormick for the Research Analysis Corporation [Ref. 1].

It was written to experiment with different methods of solving nonlinear programming problems.

A Guide to SUMT - Version 4 [Ref. 2] is the program manual which contains detailed information about the development of the code, the subroutines, options, input-output, limitations, user-supplied subroutines, data deck setup and a detailed example of the use of the program to solve a nonlinear programming problem.

The method used by SUMT to solve the original constrained optimization problem is described in detail in Chapter 8 of the book on nonlinear programming by Fiacco and McCormick [Ref. 3]. Basically, SUMT solves the constrained problem by finding the solutions of a sequence of unconstrained problems which approach the solution of the original constrained problem.

Previous versions of SUMT used different penalty functions for approximating the solution to the constrained problem but the function currently used to sequentially solve the problem is of the form:

$$P(x,r) = f(x) - r \sum_{j=1}^{m} \ln g_j(x) + \sum_{j=m+1}^{m+p} [h_j(x)]^2/r \quad ,$$

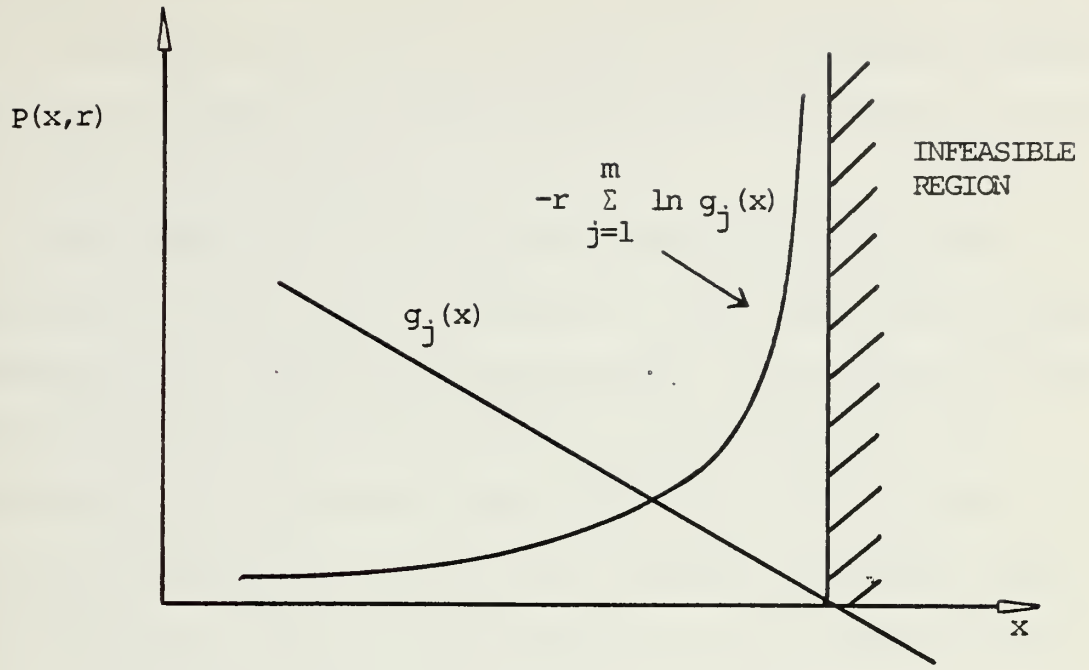
where the parameter r determines the severity of the penalty and consequently how closely the unconstrained problem solution approaches the solution to the constrained problem.

The penalty function of SUMT uses the mixed interior point-exterior point method described in detail in Chapter 4

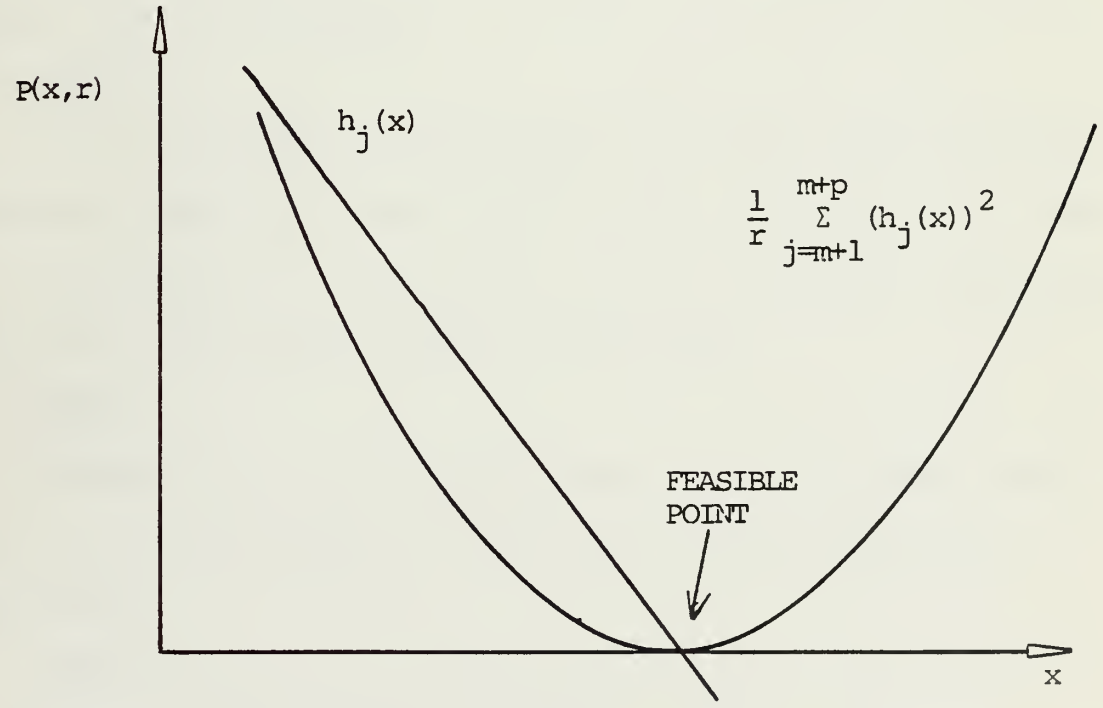
of Ref. 2. The term, $r \sum_{j=1}^{m} \ln g_j(x)$, involving the inequality constraints is from the interior point method which is also known as the barrier method. As the x vector approaches the boundary of the infeasible region for these constraints, this term approaches infinity. The term, $\sum_{j=m+1}^{m+p} [h_j(x)]^2/r$, involving the equality constraints is from the exterior point method. As the x vector moves farther away from the feasible region for these constraints, this term approaches infinity. The behavior of these two terms is shown graphically in Figure 1. How much of a penalty these terms add to the penalty function depends upon the value of the parameter r.

The solution procedure requires minimization of P(x,r) over those x satisfying the conditions $g_j(x) > 0$, j=1,...,m for $r = r_1, r_2, \ldots$ where $r_1 < r_2 < \ldots < r_k < \ldots < 0$. Under mild conditions on the original NLP problem the minima of the sequential unconstrained problems approach the solution of the original constrained problem as $r_k \to 0$. The conditions to be met for the method to solve the original problem are described in detail in the SUMT program manual [Ref. 2].

When the NLP problem is convex the SUMT program also generates a sequence of points that are feasible for an optimization problem called the dual of the original problem. The maximum function value of the dual feasible points and minimum function value of the unconstrained problem bracket the optimal solution of the constrained problem.



a. INTERIOR POINT METHOD INVOLVING INEQUALITY CONSTRAINTS



b. EXTERIOR POINT METHOD INVOLVING EQUALITY CONSTRAINTS

FIGURE 1. BEHAVIOR OF THE PENALTY TERMS

As the unconstrained problem is minimized and the number of dual feasible points found increases the difference between the two solutions decreases and the optimal solution is more accurately approximated.

SUMT makes use of the theory derived in Fiacco and McCormick's book [Ref. 3, Chapter 5] that uses the Lagrange extrapolation technique to approximate the solution of the constrained problem when more than one minimum of the unconstrained problem has been found. By using these extrapolation approximations the solution converges faster and achieves a closer approximation to the actual solution than the minima of the unconstrained problems.

The logic of the computer program taken from the SUMT manual [Ref. 2] follows.

STEP 1.

Find an initial starting point x_o within the interior feasible region such that $g_j(x) > 0$, $j=1,\ldots,m$. If such an initial point is not given or is not feasible, the program uses the optimization method to find one.

STEP 2.

Determine r_1, the initial value of r, which can either be an input parameter or a rule for choosing r_1 can be specified.

STEP 3.

Determine the minimum of the unconstrained penalty function for the current value of r.

STEP 4.

Estimate a better solution using the extrapolation technique, if possible.

STEP 5.

Computation is terminated if the stopping criteria are satisfied thus yielding an approximate solution to the original problem with accuracy depending upon the stringency of the stopping criteria.

STEP 6.

If the stopping criteria are not satisfied select $r_{k+1} < r_k$.

STEP 7.

Go to Step 3.

A simplified flow diagram of the computer logic taken from the SUMT manual [Ref. 2, p. 5] is shown in Figure 2.

The single variable search problem is involved in Step 3 of the procedure. In order to minimize the unconstrained penalty function the program chooses a search direction in n-space. It is believed that by searching in this direction from the starting point of the subproblem the penalty function will initially decrease. A one dimensional search is then made along that direction until a local minimum is found. At the solution to this one dimensional search problem a new search direction is computed and a new one dimensional search is performed. The one dimensional searches are repeated, each time with a new search direction, until the unconstrained problem for a given r is solved.

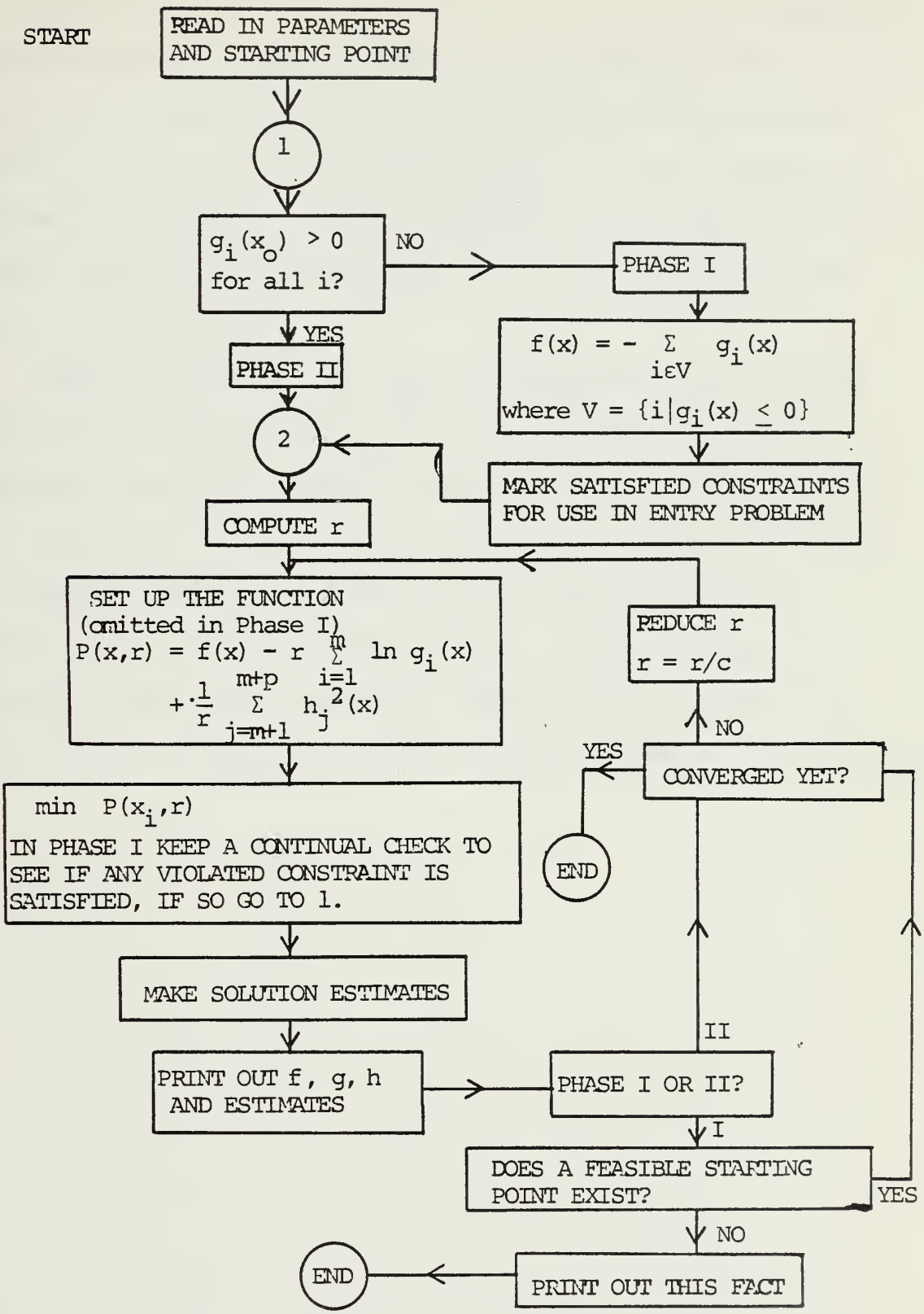


FIGURE 2. SUMT Program Flow Diagram [Ref. 2, p. 5]

SUMT allows the user to choose one of three procedures to determine the search direction — by Newton's Method using first and second partial derivatives of the unconstrained penalty function, by using the negative of the gradient of the penalty function or by a variable metric method which is taken from an algorithm of the Fiacco and McCormick book [Ref. 3, pp. 170-175].

Finding a solution to the unconstrained n dimensional problem thus requires several single variable searches. Therefore, the more efficient the single variable search is, the faster the computation time will be. Given a direction of search, the SUMT program uses the subroutine OPT to perform the single variable search. In this paper, the presently used single variable search method in the OPT subroutine is compared to other single variable search methods.

III. SINGLE VARIABLE SEARCH METHODS

Since a major portion of finding a solution to the constrained optimization problem involves single variable searches, the SUMT program was used to compare several single variable search methods.

The current OPT subroutine of SUMT uses the Golden Section search method to perform the single variable search. Two different single variable search methods were programmed to replace the current OPT subroutine. The first method uses cubic interpolation once the local minimum is bracketed to accelerate convergence to that local minimum and the second method programmed uses quadratic interpolation to accelerate convergence once three feasible points that bracket the local minimum are found. Single variable search methods by Lasdon, Fox and Ratner [Ref. 4] and Fletcher and McCann [Ref. 5] are also discussed.

The object of the single variable search is to find a scalar θ called the step length such that $x^* = x_o + \theta s$ where x^* is the x vector that corresponds to the local minimum of the penalty function along the search direction s from an initial starting point x_o. For ease of notation penalty function values are shown as a function of step length θ in each single variable search since the initial starting point x_o, the search direction s and the parameter r are constant, i.e. $P(\theta) = P(x_o+\theta*s,r)$.

The optimal step length could possibly be found by moving along the search direction in a random manner. However, this would not be a very logical way to attack the problem and would lead to many needless evaluations of the penalty function which would increase computation time.

The OPT subroutine is supplied with a search direction and an initial point from which to search. It must call other subroutines of the SUMT program for functional, gradient or Hessian evaluations and is expected to return with a penalty function value and corresponding x vector which is the local minimum for that single variable search. It is assumed that the penalty function has a finite local minimum along the given search direction.

The following subsections formulate and explain the different single variable search methods.

A. GOLDEN SECTION SEARCH METHOD

The Golden Section search method uses the limiting properties of the Fibonacci series [Ref. 6]. Fiacco and McCormick's book [Ref. 3] define the steps of the procedure used in the Golden Section method.

STEP 1.

First an upper bound step length θ_U is obtained with the lower bound step length θ_L initially equal to zero. θ_U is obtained by evaluating the penalty function at successive step lengths which are in the limiting Fibonacci ratio,

$(1+\sqrt{5})/2 \approx 1.618$, that is, $\theta_U = \sum_{i=0}^{k} (1.618)^i$, where k is the smallest integer j such that

$$P\left[\sum_{\ell=0}^{j} (1.618)^{\ell}\right] \geq P\left[\sum_{\ell=0}^{j-1} (1.618)^{\ell}\right]$$

with θ_L updated as $\sum_{\ell=0}^{j-2} (1.618)^{\ell}$.

STEP 2.

The interval bounding θ^*, the optimal step length, is reduced by computing two other step lengths within the interval (θ_L, θ_U). Their corresponding values are:

$$\theta_a = \theta_L + 0.382(\theta_U - \theta_L)$$

$$\theta_b = \theta_L + 0.618(\theta_U - \theta_L) .$$

STEP 3.

The penalty function values of the two interior step lengths, θ_a and θ_b, are then compared.

STEP 4.

If $P(\theta_a) < P(\theta_b)$, then the optimal step length which corresponds to the local minimum of the single variable search should be in the interval (θ_L, θ_b) if the penalty function is unimodal. Because of the fact that $0.382/0.618 \approx 0.618$, reassign $\theta_U = \theta_b$, $\theta_b = \theta_a$ and calculate a new θ_a as computed in Step 2. Return to Step 3.

STEP 5.

If $P(\theta_a) > P(\theta_b)$, then the optimal step size should be in the interval (θ_a, θ_U). Reassign $\theta_L = \theta_a$, $\theta_a = \theta_b$ and calculate a new θ_b as computed in Step 2. Return to Step 3.

STEP 6.

If the penalty function values at both interior step lengths are equal, reassign $\theta_L = \theta_a$, $\theta_U = \theta_b$ and calculate a new θ_a and θ_b as computed before by returning to Step 2.

STEP 7.

When the stopping criteria is satisfied as explained in subsection III.D, θ^* is approximated by $(\theta_L + \theta_U)/2$ yielding an x vector $x^* = x_o + \theta^* S$ which corresponds to an approximated local minimum $P(\theta^*)$ of the single variable search.

A flow diagram of the Golden Section method presently used in the OPT subroutine is shown in Figure 3. Certain parts of the computer code that were the same in each of the OPT subroutines are discussed in subsection III.D.

The Golden Section OPT subroutine initially updates θ_L as it finds more feasible points by increasing the step length until it finds a point which is infeasible or where the penalty function value is larger than the penalty function value at the previous feasible point. If it finds an infeasible step length, the penalty function is assigned a very high value (10^{36}), the next to the last feasible step length is assigned θ_L and the last feasible step length assigned as the lower interior step length θ_a. If the first

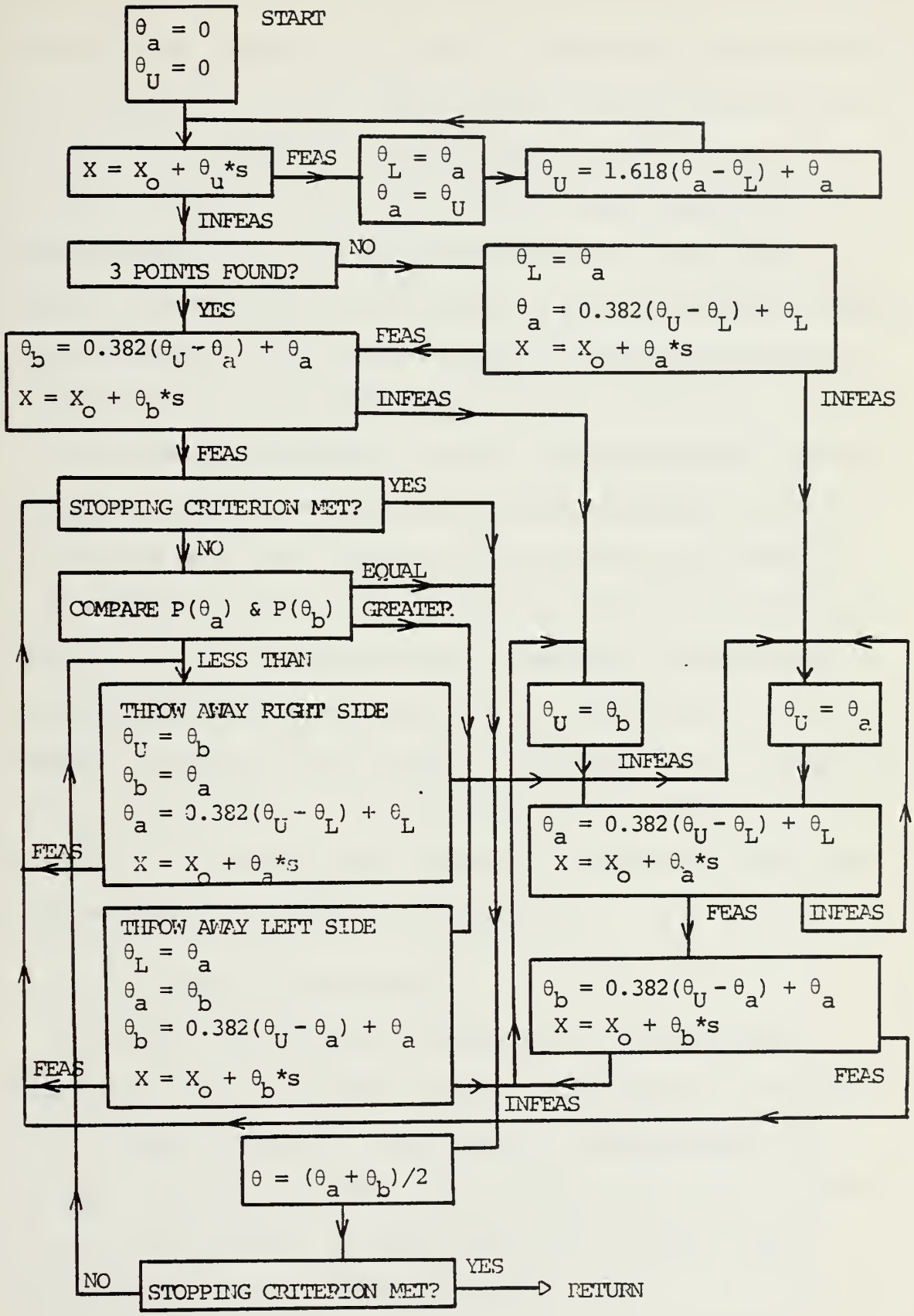


FIGURE 3. Golden Section OPT Subroutine

step length from the initial point is infeasible, the lower interior step length $\theta_a = 0.382\ \theta_U$, since only one feasible step length ($\theta_L = 0$) has been found. OPT then uses θ_U and θ_a to compute the upper interior step length $\theta_b = \theta_a + 0.382(\theta_U - \theta_a)$. The optimal step length is bracketed and the subroutine continues to reduce the interval of uncertainty by comparing penalty function values of the interior step lengths until it finds two values or an interval that satisfies the stopping criteria.

In finding the minimum of each single variable search, the Golden Section method only requires penalty function evaluations which are compared to show where the penalty function decreases to a local minimum and then increases to infinity as the infeasible region boundary is approached. The factor by which the interval of uncertainty is reduced remains constant in this method. Therefore, any information about the behavior of the penalty function other than the fact that the optimal step length is somewhere within the interval of uncertainty is disregarded.

B. CUBIC INTERPOLATION METHOD

The cubic interpolation method was programmed for comparison with the Golden Section and quadratic interpolation methods. Basically, this method approximates the penalty function by a cubic fit and then finds the minimum of the cubic equation to approximate the optimal step length of the single variable search.

In order to approximate the penalty function by a cubic fit in the single variable search, two feasible step lengths which bracket the local minimum along with their directional derivative values are needed. This method actually consists of two distinct parts – the bracketing section to find the two feasible points and the cubic interpolation section to find a minimum of the cubic function. The single variable search method is supplied with an initial starting point x_o, the penalty function value at x_o [P(0)], the gradient of the penalty function at x_o [∇P(0)] and a search direction s.

The steps of the cubic interpolation method follow.

STEP 1.

Since the directional derivative at the starting point P'(0) is negative, by finding a step length that has a corresponding positive directional derivative, the local minimum along the given direction of search will be bracketed if the penalty function is unimodal.

The step length is increased to find an upper step length θ_U until an n is found such that

$$P'\left[\left(\sum_{i=1}^{n} 2^i\right) - 1\right] > 0 \quad .$$

If, while initially increasing the step length, an infeasible point is encountered, the increment by which the last step length was increased is halved and subtracted from the current step length. This is continued until a step length that is feasible is found. If its directional derivative is

negative, the increment is increased by one half the increment length and added to the current step length. This is continued until a step length with a positive directional derivative is found.

The lower step length θ_L is always the last feasible step length encountered with a negative directional derivative.

STEP 2.

Once two step lengths have been found that bracket the local minimum along the search direction, the cubic equation is used to compute a step length which corresponds to the minimum of the cubic fit. This step length is found by the equation:

$$\theta = \theta_U - (\theta_U - \theta_L) \left[\frac{P'(\theta_U) + U_2 - U_1}{P'(\theta_U) - P'(\theta_L) + 2U_2} \right]$$

where

$$U_1 = P'(\theta_L) + P'(\theta_U) - 3 \left[\frac{P(\theta_L - P(\theta_U)}{\theta_L - \theta_U}\right]$$

$$U_2 = [U_1^2 - P'(\theta_L)P'(\theta_U)]^{1/2}$$

STEP 3.

$P'(\theta)$ is evaluated and if it is negative then reassign $\theta_L = \theta$ and the left side is discarded. If $P'(\theta)$ is positive then reassign $\theta_U = \theta$ and the right side is discarded.

STEP 4.

If the stopping criteria is satisfied, the two penalty function values, $P(\theta)$ and $P(\theta_L)$ or $P(\theta)$ and $P(\theta_U)$, depending upon $P'(\theta)$, are compared and the smallest one is returned as the local minimum $P(\theta^*)$ of the single variable search with its corresponding x vector, $x_o + \theta^* s$.

STEP 5.

If the stopping criteria is not satisfied return to Step 2.

A flow diagram of the cubic interpolation OPT subroutine is shown in Figure 4. The parts of this subroutine that are the same in the other OPT subroutines are discussed in subsection III.D.

The cubic interpolation method uses more information at each feasible step length to find the local minimum of the single variable search than the Golden Section or quadratic interpolation methods. It requires a gradient evaluation for the penalty function at each feasible step length in order to compute the directional derivative. A gradient evaluation may require many function evaluations depending upon the number of variables and constraints in the original constrained problem. However, with this extra information the local minimum should be found by evaluating fewer step lengths in the cubic interpolation section of this method. As the value of the parameter r decreases with each unconstrained problem, the penalty function rises more abruptly as the infeasible region boundary is approached.

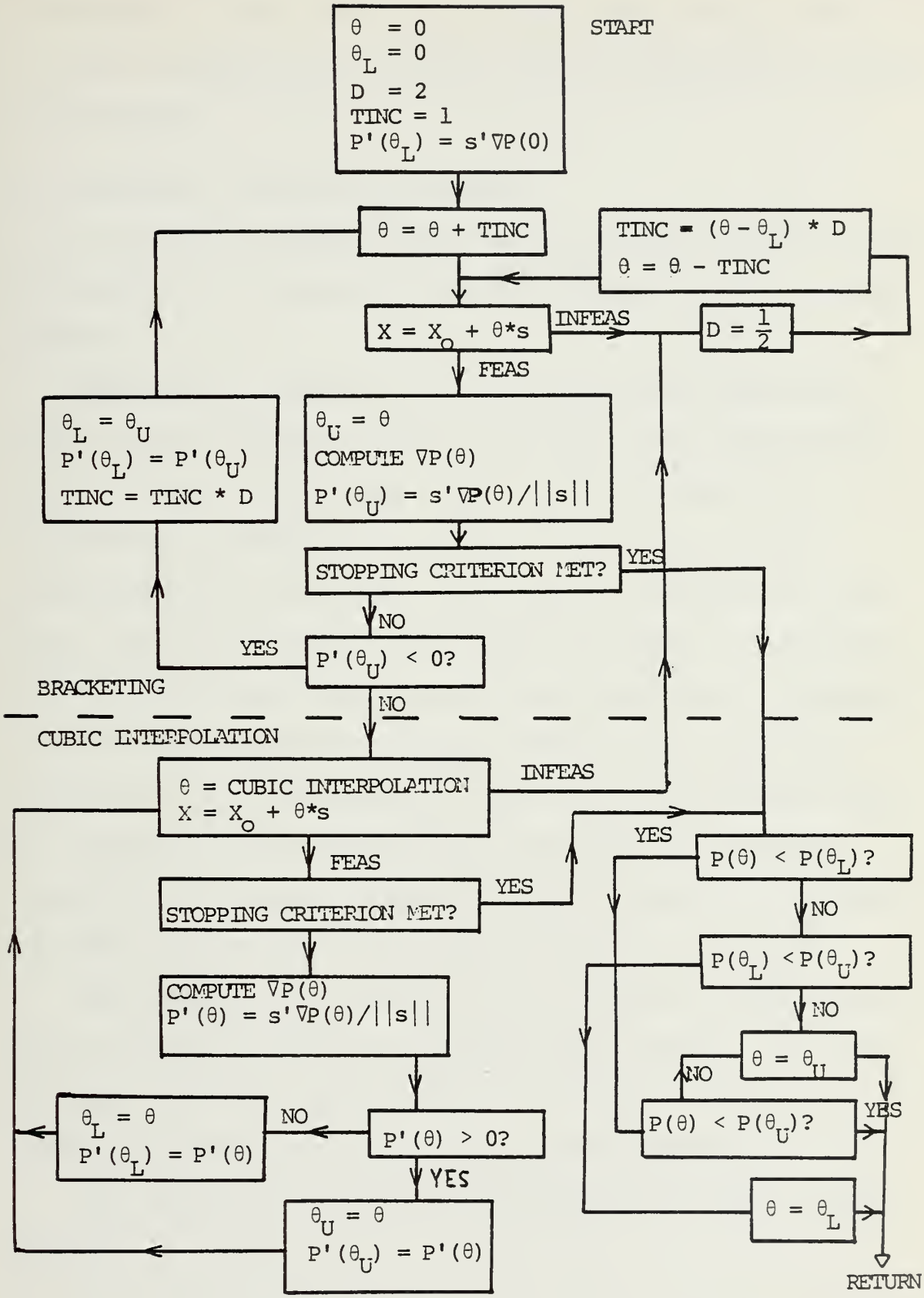


FIGURE 4. Cubic Interpolation OPT Subroutine

This makes it more difficult to find a step length with a positive directional derivative which is required to bracket the minimum.

C. QUADRATIC INTERPOLATION METHOD

A quadratic interpolation single variable search was also programmed for comparison with the two previously discussed methods.

This method requires three feasible step lengths and their penalty function values which obey the relationship $P(\theta_L) > P(\theta_M) < P(\theta_U)$ where $\theta_L < \theta_M < \theta_U$. With this information a quadratic fit is made with its minimum approximating the minimum of the penalty function for the given search direction. The penalty function is evaluated at the interpolated step length which minimizes the quadratic function and compared to $P(\theta_M)$ allowing the interval of uncertainty to be decreased. The process is repeated until the minimum of the quadratic fit approximates the local minimum of the penalty function or the interval of uncertainty becomes small enough to satisfy the stopping criteria.

This method also consists of two sections — the bracketing section and the quadratic interpolation section. The single variable search is supplied with an initial x vector x_o, its penalty function value P(0) and a search direction s.

The steps of the quadratic interpolation search method follow.

STEP 1.

The step length is increased from the initial starting point to find an upper step length θ_U until an n is found such that

$$P\left[\left(\sum_{i=1}^{n} 2^i\right) - 1\right] > P\left[\left(\sum_{i=1}^{n-1} 2^i\right) - 1\right] \quad .$$

If, while initially increasing the step length, an infeasible step length is encountered, the increment by which the last step length was increased is halved and subtracted from the current step length. This is continued until a feasible step length is found. If its functional value is less than $P(\theta_L)$, the increment is increased by one half the increment length. This is continued until a feasible step length is found that has a larger penalty function value than the previous step length. Before this is repeated, reassign $\theta_L = \theta_M$ and $\theta_M = \theta_U$, always ensuring that $P(\theta_L) > P(\theta_M)$.

If only one feasible step length is found in obtaining the upper step length, where in this case $P(\theta_U) > P(\theta_L)$ (which indicates the minimum is bracketed), the last increment is halved and subtracted from the upper step length which yields the interior step length θ_m.

STEP 2.

Now that the local minimum is bracketed, the quadratic function is used to find a step length where the derivative of the quadratic fit vanishes.

$$\theta = \frac{1}{2} \frac{b_{23}P(\theta_L) + b_{31}P(\theta_M) + b_{12}P(\theta_U)}{a_{23}P(\theta_L) + a_{31}P(\theta_M) + a_{12}P(\theta_U)}$$

where $a_{ij} = \theta_i - \theta_j$, $b_{ij} = \theta_i^2 - \theta_j^2$,

$i,j = 1,2,3$ where 1 = L, 2 = M, and 3 = U.

STEP 3.

If $\theta > \theta_M$, then the left side is discarded and reassign $\theta_L = \theta_M$ and $\theta_M = \theta$. If $\theta < \theta_M$, then the right side is discarded and reassign $\theta_U = \theta_M$ and $\theta_M = \theta$.

STEP 4.

$P(\theta_M)$ is evaluated and if the stopping criteria are satisfied, the smallest of $P(\theta_L)$, $P(\theta_M)$ and $P(\theta_U)$ is returned as $P(\theta^*)$ with the corresponding x vector, $x_o + \theta^* s$.

STEP 5.

If the stopping criteria are not met, return to Step 2.

A flow diagram of the quadratic interpolation method OPT subroutine is shown in Figure 5.

The quadratic interpolation method only requires function evaluations to perform the single variable search. It uses the knowledge of three feasible step lengths and their penalty function values to approximate where the local minimum of the single variable search is located. However, it must find an upper feasible step length that is within the interval where the penalty function is increasing to infinity and also this feasible step length must have a penalty function value that is greater than that of the interior

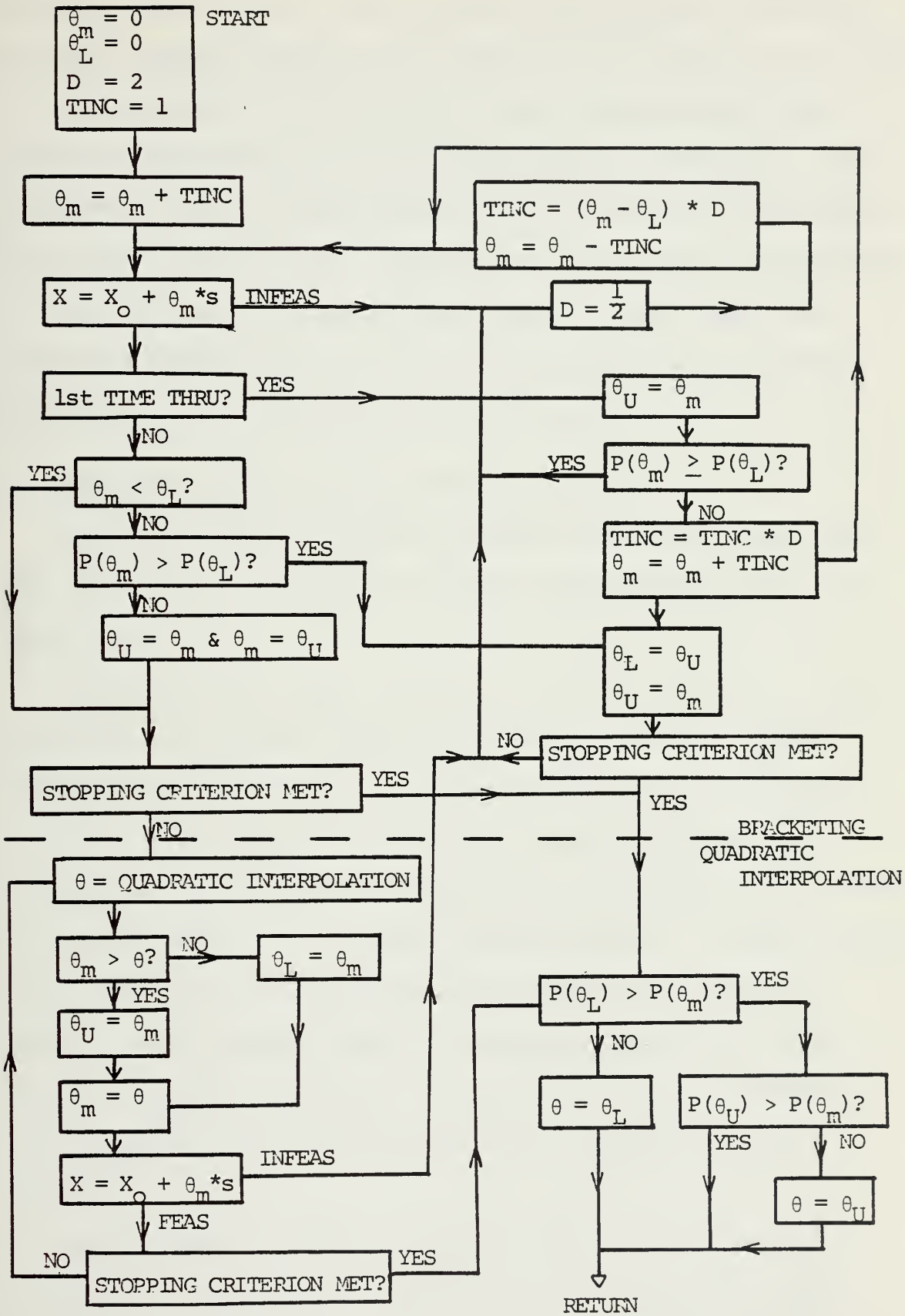


FIGURE 5. Quadratic Interpolation OPT Subroutine

feasible step length. This causes the interval containing possible feasible upper step lengths to be reduced even more. As the parameter r decreases for each unconstrained sub-problem, the penalty function increases more abruptly from the minimum as the infeasible region boundary is approached. This also increases the difficulty in bracketing the minimum.

Once the three feasible step lengths are found, the quadratic interpolation can converge to the local minimum thereby solving the single variable search.

D. SECTIONS COMMON TO THE DIFFERENT METHODS

Some parts of the three different OPT subroutines were made similar so that they could be compared under the same conditions.

Essentially the same stopping criteria was used in each subroutine. When the ratios of the two x vectors that bracketed the minimum or ratios of their penalty function values were within 10^{-7} of one, the subroutine returned to the program with a local minimum and a corresponding x vector as the solution to the single variable search. These criteria were checked in the Golden Section method whenever two interior step lengths had been computed and in the interpolation methods both after the minimum had been bracketed and also after each interpolated step length had been computed.

In the three different methods, the subroutines also contained counters which would cause the search to stop

after ten interpolations in the cubic interpolation method, twenty interpolations in the quadratic interpolation method or twenty-five feasible points in the Golden Section method. In the test problems that were run these counters were never reached, thereby, never satisfying this stopping criterion.

In the cubic interpolation method another stopping criterion was used along with the ratio tests. When the cosine of the angle between the gradient of the penalty function and the search direction vector was less than 10^{-7} the subroutine returned with a local minimum and x vector.

It was stated earlier that it was hoped that the penalty function would initially decrease in the given search direction from the initial starting point. In the three different methods, if the ratio of any element of the current x vector and the corresponding element in the initial x vector came within 10^{-7} of one, the single variable search was restarted in the opposite search direction. In the interpolation methods if after one hundred tries the minimum was not bracketed or the step length had been decreased more than twenty consecutive times, the search direction was also reversed and the single variable search was restarted.

The three different OPT subroutines called the subroutine EVALU for penalty function evaluations. EVALU would return back a variable after each call up which would show if the step length was feasible. The cubic interpolation method called the subroutine GRAD for the gradient of the penalty function when it was needed to determine the directional derivative.

E. OTHER SINGLE VARIABLE SEARCH METHODS

Two other single variable search methods were also looked at but were not programmed because of their complexity and their need for changing other portions of the SUMT program than just the OPT subroutine.

1. Lasdon, Fox, and Ratner Method

The single variable search developed by Lasdon, Fox and Ratner [Ref. 4] uses a penalty function that doesn't contain equality constraints and the penalty term for the inequalities is also different than the SUMT penalty function. The unconstrained minimization problem is:

$$\text{minimize } P(x,r) = F(x) + r \sum_{i=1}^{m} \frac{1}{G_i(x)}$$

where F is the objective function and the G_i's are the inequality constraints. This model could be modified to handle equality constraints and a penalty function like that used in SUMT.

The single variable search method is in three distinct stages — linear approximation, quadratic approximation and cubic interpolation. Linear approximations are made for the objective function and each constraint using $F(0)$, $F'(0)$, $G_j(0)$ and $G_j'(0)$. A step length (θ_1) which minimizes the penalty function consisting of the linear approximations is then calculated. The information used to make the linear approximations is then used along with $F(\theta_1)$, $F'(\theta_1)$, $G_j(\theta_1)$ and $G_j'(\theta_1)$ to make quadratic approximations of the

objective function and constraints. The penalty function consisting of quadratic approximations is then minimized by computing θ_2. If the directional derivative of the original penalty function is positive at θ_2, cubic interpolation is applied yielding θ^*, the solution to the single variable search. If the stopping criterion is satisfied during any stage, the single variable search is terminated yielding a local minimum and a solution to the search.

The following steps outline the Lasdon, Fox and Ratner procedure.

STAGE 1.

The objective function and inequality constraints of the original constrained problem are approximated by using the given values at the initial starting point.

$$f_1(\theta) = F(0) + F'(0)$$

$$g_{1j}(\theta) = G_j(0) + G_j'(0) \qquad j = 1,\ldots,m.$$

The approximating function for $P(\theta)$ is

$$p_1(\theta) = f_1(\theta) + r \sum_{j=1}^{m} \frac{1}{g_{1j}(\theta)}$$

The smallest positive zero of the linear approximations of the inequality constraints is found by taking the smallest $-G_j(0)/G_j'(0)$ over all j. This value is designated the upper step length θ_U.

The step length corresponding to the minimum of the approximated penalty function is found by doing four or five

Newton's method iterations using $\frac{1}{2}\ \theta_U$ as the starting point.

$$\theta_1 = \frac{1}{2}\ \theta_U - \frac{p_1'(\frac{1}{2}\ \theta_U)}{p_1''(\frac{1}{2}\ \theta_U)}$$

The result of this stage is a step length that approximates the optimal step length of the single variable search. If the stopping criterion, explained following Stage 3, is not satisfied proceed to Stage 2.

STAGE 2.

The same procedure is performed in this stage as in Stage 1 with the exception that quadratic approximations to the objective function and constraints are made instead of linear approximations. The second stage is entered with values of $F(\theta_1)$ and $G_j(\theta_1)$ along with the information used to make the linear approximations. The quadratic approximations to the objective function and constraints are

$$f_2(\theta) = a\theta^2 + b\theta + c$$

where $$a = \frac{F(\theta_1) - b\theta_1 - c}{\theta_1^2}$$

$$b = F'(0)\ ,\quad c = F(0)$$

and

$$g_{2j}(\theta) = d_j\theta^2 + e_j + f_j \qquad j = 1,\ldots,m$$

where $$d_j = \frac{G_j(\theta_1) - e_j\theta_1 - f_j}{\theta_1^2}$$

$$e_j = G_j'(0)\ ,\quad f_j = G_j(0).$$

If $G_j(\theta_1)$ is infeasible for some j, then the smallest positive root over all j of

$$\frac{-e_j \pm \sqrt{e_j^2 - 4d_j f_j}}{2dj} = \theta_{2U}$$

is used as the starting step length for using Newton's method to minimize the approximated penalty function

$$p_2(\theta) = f_2(\theta) + r \sum_{i=1}^{m} \frac{1}{g_{2j}(\theta)} .$$

The step length which minimizes $p_2(\theta)$ should be found after four or five iterations of Newton's method.

$$\theta_2 = \theta_1 - \frac{p_2'(\theta_1)}{p_2''(\theta_1)}$$

If θ_1 is infeasible, use $\frac{1}{2}\ \theta_{2U}$ in Newton's method. If the stopping criterion is not satisfied after finding θ_2 continue to Stage 3.

STAGE 3.

Direct cubic interpolation of the penalty function is used in this stage. If $P'(\theta_2)$ is positive, the minimum is bracketed and cubic interpolation can be performed to find a step length which corresponds to the minimum of the penalty function. If the minimum has not been bracketed, that is, if $P'(\theta_2) < 0$, a bracketing procedure like the one used in the cubic interpolation method of subsection III.B

must be performed to find a step length that has a positive directional derivative. Once this step length is found, cubic interpolation is used to find a step length which minimizes the cubic function. If the new step length does not satisfy the stopping criteria, additional cubic interpolations are made using the two points which most closely bracket the minimum of the penalty function.

When two penalty function values are nearly equal or when the interval between step lengths that bracket the minimum becomes very small, an optimal step length is determined by

$$\theta^* = \frac{P'(\theta_a)\theta_b - P'(\theta_b)\theta_a}{P'(\theta_a) - P'(\theta_b)} \quad .$$

where θ_a is the lower step length and θ_b is the upper step length ($\theta_a < \theta^* < \theta_b$).

The stopping criterion used in this method is the cosine test $|\cos \phi| = \frac{|s^T \nabla P|}{||s|| ||\nabla P||}$, where ϕ is the angle between the search direction vector and gradient of the penalty function. When the cosine of the angle is less than 10^{-2}, the stopping criterion is satisfied.

The authors found that in using this method on test problems, the stopping criterion was often satisfied at the end of Stage 2. In their comparisons of this method with a cubic interpolation method, they found that it performed the single variable search much more efficiently [Ref. 4, p. 295].

This method requires gradients of the objective function and constraints. In the SUMT program these values are not stored so implementing this method would require more storage or more gradient evaluations along with changes in other subroutines and/or addition of new subroutines.

2. Fletcher-McCann Method

The Fletcher-McCann method [Ref. 5, pp. 210-212] uses an approximation of the original unconstrained penalty function to find the local minimum of the single variable search. The penalty function approximation used is of the form

$$T(\theta) = a + b\theta + c\theta^2 + \frac{d}{\theta_U - \theta} ,$$

where the parameters a, b, c, d and θ_U are determined by using objective function and constraint information at two feasible step lengths. The authors felt that an approximation of this type which goes to infinity at the barrier $\theta = \theta_U$ would fit the penalty function better than a polynomial approximation which goes to infinity only as $\theta \to \infty$. θ_U is an estimate of the intersection of the search direction s with the boundary of the infeasible region. The approximated penalty function's minimum is found by applying Newton's method to find a step length which corresponds to the local minimum along the direction of search.

Since the explanation of this method by the authors was rather sketchy, certain parts could be interpreted

differently and only resolved by testing the method on different problems. An explanation of the notation for this method is first given.

$$F(\theta) = f(\theta) + \frac{1}{r} \sum_{j=m+1}^{m+p} h_j^2(\theta) \approx a + b\theta + c\theta^2$$

$$G(\theta) = -r \sum_{j=1}^{m} \ln g_j(\theta) \approx \frac{d}{(\theta_U - \theta)}$$

$$F'(\theta) = \frac{s^T \nabla f(\theta) + \frac{2}{r} \sum_{j=m+1}^{m+p} h_j(\theta) s^T \nabla h_j(\theta)}{||s||} \approx b + c\theta$$

$$G'(\theta) = -\frac{r}{||s||} \sum_{j=1}^{m} \frac{s^T \nabla g_j(\theta)}{g_j(\theta)} \approx \frac{d}{(\theta_U - \theta)^2}$$

The single variable search is supplied with an initial point, functional and gradient information at that point and a search direction. The steps of the method follow.

STEP 1.

A feasible step length in the direction of search is needed along with the initial point in order to estimate the parameters a, b, c, d and θ_U. A unit step length is initially taken in trying to find a feasible step length. If it is not feasible, the step length is progressively halved until a feasible step length is found. The lower and upper feasible step lengths are written as θ_0 and θ_1, respectively. At the start of the search $\theta_0 = 0$. Function and gradient evaluations

are made at the second feasible step length in order to compute F_1, G_1, F_1' and G_1' (shortened notation for $F(\theta_1)$, $G(\theta_1)$, etc.).

By using the information at the two feasible step lengths, simultaneous equations are solved to determine the parameters. The parameter values are

$$c = \frac{F_0' - F_1'}{2(\theta_0 - \theta_1)}$$

$$b = F_0' - 2c\theta_0$$

$$U = \frac{(G_0'\theta_0 - G_1'\theta_1) \pm (\theta_1 - \theta_0)\sqrt{G_0'G_1'}}{G_0' - G_1'}$$

$$d = G_0'(\theta_U - \theta_0)^2 \quad .$$

The parameter θ_U is taken as the smallest value of two computed values which is greater than θ_1.

STEP 2.

Once the parameters are determined, the equation for the minimum of T which is

$$T' = 0 = b + 2c\theta + \frac{d}{(\theta_U - \theta)^2}$$

is solved by Newton's method using the midpoint between the two feasible step lengths as the starting point.

$$\theta = \left(\frac{\theta_1 - \theta_0}{2}\right) - \left[\frac{T'\left(\frac{\theta_1 - \theta_0}{2}\right)}{T''\left(\frac{\theta_1 - \theta_0}{2}\right)}\right]$$

where $T'' = 2c + \frac{2}{3}\frac{d}{(\theta_U - \theta)^3}$.

STEP 3.

The values of $\nabla P(\theta)$ are computed and if the stopping criterion is satisfied, the single variable search is terminated with θ as the optimal step length.

STEP 4.

If the stopping criterion is not satisfied, a new step length is computed using Newton's method in Step 2.

The authors of this method stated that different default actions would be necessary in cases where the interpolation failed or was unsatisfactory. For example, if for the feasible step lengths, θ_0 and θ_1, a value of θ_U could not be determined, it would be necessary to use another feasible step length to determine where the intersection of the search direction and the boundary of the infeasible region was. In later stages of the minimization the authors said that this method sometimes failed and a quadratic interpolation would have to be used.

Implementing this method in the SUMT program would also require more storage or more gradient evaluations along with modifications to other subroutines and/or additional subroutines.

No comparisons are made of the Lasdon, Fox and Ratner or Fletcher-McCann methods to the other three methods since they are only discussed and not programmed.

IV. TEST PROBLEMS

A. TEST PROBLEM ORIGIN

The problems used to compare the three different single variable search methods programmed as OPT subroutines were taken from problems used in a thesis by Lt. J. Waterman, USN, which compared three different nonlinear programming codes [Ref. 7]. The SUMT program and problem data decks were the same as those used by Waterman except for the Golden Section OPT subroutine being replaced by the cubic and quadratic interpolation methods.

The structure and degree of difficulty among the seven test problems are quite varied and represent a sample of some real world problems. The number of variables and constraints ranged from 9 to 100 and 2 to 20 respectively. The problems contain combinations of linear, nonlinear, equality and inequality constraints.

The following subsection contains the descriptions of the test problems as presented in Waterman's thesis. Constants in each problem are represented by a, b, c, d, e, L, m, u and s unless otherwise specified.

B. TEST PROBLEM DESCRIPTION

Problem 1 was an example of determining the chemical composition of a complex mixture under conditions of chemical equilibrium. It contained 45 independent variables and 16 linear equality constraints.

$$\text{minimize} \quad f(x) = \sum_{k=1}^{7} \left[\sum_{j=1}^{h_k} x_{jk}\left(c_{jk} + \ln \frac{x_{jk}}{\sum_{j=1}^{h_k} x_{jk}}\right) \right]$$

$$\text{subject to} \quad h_i(x) = \sum_{k=1}^{7} \left(\sum_{j=1}^{h_k} E_{ijk} x_{ijk} \right) - b_i = 0, \qquad i = 1,\ldots,16$$

$$x_{jk} \geq 0 \qquad j = 1,\ldots,h_k \qquad k = 1,\ldots,7.$$

Problem 2 was formulated by the Shell Development Company. It consisted of 15 variables and 5 nonlinear equality constraints.

$$\text{maximize} \quad f(x) = \sum_{i=1}^{10} b_i x_i - \sum_{j=1}^{5} \sum_{i=1}^{5} yz - 2 \sum_{j=1}^{5} d_j z^3$$

$$\text{where } y = c_{ij} x_{(10+i)} \quad \text{and} \quad z = x_{(10+j)}$$

$$\text{subject to} \quad g_j(x) = 2 \sum_{i=1}^{5} y + 3 d_j z^2 + e_j - \sum_{i=1}^{10} a_{ij} x_i \geq 0$$

$$x_i \geq 0 \qquad i = 1,\ldots,15$$

Problem 3 was to maximize the area of a hexagon in which the maximum diameter was unity. The problem had 9 independent variables, 13 nonlinear inequality constraints and a lower bound of 0 for x_9.

Maximize: $f(x) = .5(x_1x_4 - x_2x_3 + x_3x_9 - x_5x_9 + x_5x_8 - x_6x_7)$

subject to:

$$1 - x_3^2 - x_4^2 \geq 0$$

$$1 - x_4^2 \geq 0$$

$$1 - x_5^2 - x_6^2 \geq 0$$

$$1 - (x_1 - x_5)^2 - (x_2 - x_6)^2 \geq 0$$

$$1 - (x_1 - x_5)^2 - (x_2 - x_6)^2 \geq 0$$

$$1 - (x_1 - x_7)^2 - (x_2 - x_8)^2 \geq 0$$

$$1 - (x_3 - x_5)^2 - (x_4 - x_6)^2 \geq 0$$

$$1 - (x_3 - x_7)^2 - (x_4 - x_8)^2 \geq 0$$

$$1 - x_7^2 - (x_8 - x_9)^2 \geq 0$$

$$x_1x_4 - x_2x_3 \geq 0$$

$$x_3x_9 \geq 0$$

$$-x_5x_9 \geq 0$$

$$x_5x_8 - x_6x_7 \geq 0$$

$$x_9 \geq 0 \quad .$$

Problem 4 was probably the most difficult of the test problems. It included a linear objective function, 24 variables, 12 nonlinear equality, 2 linear equality and 6 non-linear inequality constraints. The variables had to also be zero or positive.

$$\text{minimize:} \quad f(x) = \sum_{i=1}^{24} a_i x_i$$

subject to:

$$h_i(x) = \frac{x_{(i+12)}}{b_{(i+12)} \sum_{j=13}^{24} \frac{x_j}{b_j}} - \frac{c_i x_i}{40 b_i \sum_{j=1}^{12} \frac{x_j}{b_j}} = 0, \quad i = 1,\ldots,12$$

$$h_{13}(x) = \sum_{i=1}^{24} x_i - 1 = 0$$

$$h_{14}(x) = \sum_{i=1}^{12} \frac{x_i}{d_i} + f \sum_{i=13}^{24} \frac{x_i}{b_i} - 1.671 = 0$$

where $f = 142.224$

$$h_{(i+14)}(x) = \frac{-(x_i + x_{(i+12)})}{\sum_{j=1}^{24} x_j} + e_i \geq 0 \ , \quad i = 1,2,3$$

$$h_{(i+14)}(x) = \frac{-(x_{(i+3)} + x_{(i+15)})}{\sum_{j=1}^{24} x_j} + e_i \geq 0, \quad i = 4,5,6$$

$$x_i \geq 0, \quad i = 1,\ldots,24 \ .$$

Problem 5 was a weapon assignment problem with 100 variables, a nonlinear objective function, 12 linear constraints and zero lower bounds for the variables.

$$\text{minimize:} \quad f(x) = \sum_{j=1}^{20} u_j \left(\prod_{i=1}^{5} a_{ij}^{x_{ij}} - 1 \right)$$

subject to:

$$\sum_{i=1}^{5} x_{ij} - b_j \geq 0 \qquad j = 1,6,10,14,15,16,20$$

$$- \sum_{j=1}^{20} x_{ij} + c_i \geq 0 \qquad i = 1,\ldots,5$$

$$x_{ij} \geq 0 \qquad i = 1,\ldots,5, \quad j = 1,\ldots,20 \ .$$

Problem 6 was adapted from an inventory model where the x_i, $i = 1,\ldots,50$, represent the reorder quantity for 50 inventory items and x_i, $i = 51,\ldots,100$, represent the reorder points for the same 50 items. It contained 100 variables, 1 linear and 1 nonlinear inequality constraint, and 50 lower bounds on the variables.

$$\text{minimize:} \quad f(x) = \sum_{i=1}^{50} \frac{B_i(x_i + 50)}{x_i}$$

where

$$B_i(x_i + 50) = \frac{1}{2}(s_i^2 + d_i^2)\,\Phi\left(\frac{d_i}{s_i}\right) - \frac{s_i d_i}{2}\,\phi\left(\frac{d_i}{s_i}\right),$$

$$d_i = x_{(i+50)} - m_i \quad , \quad \phi(x) = \frac{1}{\sqrt{2\pi}}\, e^{-x^2/2}$$

and $$\Phi(r) = \int_t^x \phi(x)\, dx \, .$$

subject to:

$$200{,}000 - \sum_{i=1}^{50} c_i\left(\frac{x_i}{2} + x_{(i+50)} - m_i\right) \geq 0$$

$$300 - \sum_{i=1}^{50} \frac{L_i}{x_i} \geq 0$$

$$x_i \geq 0 \quad , \quad i = 1,\ldots,50.$$

Problem 7 was an entropy model. There were 46 population centers connected by a transportation network. Using a congestion cost function the model yields an equilibrium solution that identifies nodal populations as entropic functions of the total cost of the journey to work. The problem contained 46 variables, all of which have a zero lower bound, 1 nonlinear inequality and 1 linear equality constraint.

$$\text{minimize:} \quad f(x) = \sum_{i=1}^{46} \frac{x_i}{500} \left(\ln \frac{x_i}{500}\right)$$

$$\text{subject to:} \quad 500 - \sum_{i=1}^{46} x_i = 0$$

$$10000 - \sum_{i=1}^{46} c_i y_i + a d_i y_i^{\,b} \geq 0$$

where $y_i = x_i + \sum_{j \in A(i)} x_j$, A(i) consists of all the arcs that converge directly and indirectly upon node i, and

$$x_i \geq 0 \qquad i = 1,\ldots,46 \ .$$

The preceding problem descriptions are only meant to give a feeling of the kind of test problems used and their structure. Detailed information of how each problem was set up, the constant values, initial starting points and where the problems specifically originated is contained in Ref. 5.

V. TEST RESULTS AND COMPARISONS

A. PROGRAMMING AND TESTING PROCEDURE

The cubic interpolation method was first programmed as a program that did a single variable search when given a penalty function, its gradient, an initial starting point and a search direction. After it had been debugged it was converted into an OPT subroutine. After the subroutine was debugged using a couple of the less difficult test problems, OPT was converted into the quadratic interpolation method by modifying the bracketing procedure and changing the interpolation from a cubic function to a quadratic function.

When the quadratic interpolation method had been debugged, each problem was run through the SUMT program three times, using the three different search methods. As discussed earlier, the stopping criteria was made essentially the same in each of the OPT subroutines so that the only thing that varied in solving each test problem was the single variable search method.

In the process of running the other test problems, minor debugging changes had to be made in the interpolation methods. After all the changes had been incorporated, a final run of each problem with each method was made which produced the results used in the comparisons. Each method found the same solution to each problem to within six significant figures.

B. COMPARISON CRITERIA

Perhaps the best way to compare the different search methods is to compare the computation time required to find the solution to the problem. However, because of computer interactions, computation time may vary as much as twenty five per cent when the same problem is run at two different times. Computation times were computed for each problem to see if a trend could be seen between the methods.

Counters were inserted in the SUMT program so that the number of single variable searches performed and the number of functional and gradient evaluations needed to find the solution could be counted.

Results of the test problem runs are shown in Table I. In running problem 7 with the interpolation methods it was necessary to change the factor by which the increment was multiplied for step length increase or reduction to 1.618 vice 2 and 0.618 vice 0.5 respectively. This change was needed because a feasible starting point was found in the bracketing procedures which the SUMT program could not solve. By changing the factors to the same as those in the Golden Section method, the same feasible starting point was used to solve the problem.

C. COMPARISONS

Theoretically, each single variable search should reach the same solution for all three methods. However, because of the tolerances allowed in the stopping critera, the

TABLE I. TEST PROBLEM RESULTS

Problem		Golden Section	Cubic	Quadratic
1	TIME	105.8	105.7	94.8
	SVS	86	89	75
	F	15487	6171	10149
	G	1462	7497	1275
2	TIME	11.4	9.6	8.3
	SVS	63	53	53
	F	4354	1574	2606
	G	378	1704	318
3	TIME	4.8	5.6	4.0
	SVS	39	38	40
	F	8606	3302	5862
	G	613	3358	628
4	TIME	395.3	404.1	396.1
	SVS	163	151	174
	F	44255	13922	30048
	G	92.27	97319	98694
5	TIME	103.7	97.9	97.0
	SVS	20	19	19
	F	2683	1250	1991
	G	296	1415	271
6	TIME	165.4	145.9	155.5
	SVS	36	34	35
	F	2320	836	978
	G	104	817	107
7	TIME	78.8	80.3	67.8
	SVS	20	20	17
	F	865	307	397
	G	1993	2257	1708

TIME = Computation time
SVS = # single variable searches
F = # function evaluations
G = # gradient evaluations

solutions were slightly different at the conclusion of each single variable search. This meant that at the start of the next single variable search the search direction would be slightly different also. As several single variable searches were performed in each problem, the differences accumulated and this explains why for some of the problems it takes a different number of searches for each method to reach the same solution. This also explains the difference in the number of function and gradient evaluations required for each problem for the Golden Section and quadratic interpolation methods.

Table II shows the normalized results for the time per single variable search, function evaluations per single variable search and gradient evaluations per single variable.

As was expected, the number of gradient evaluations per single variable search is essentially the same for Golden Section and quadratic interpolation methods. The only thing that can be compared between the three methods is the computation time per single variable search. Function evaluations per single variable search can only be used as a measure of effectiveness for Golden Section and quadratic interpolation since the cubic interpolation also requires gradient evaluations and fewer function evaluations.

In looking at the times per single variable search the quadratic interpolation was faster than the Golden Section and cubic interpolation methods 5 out of 7 and 4 out of 7 test problems respectively. Cubic interpolation was faster

TABLE II. NORMALIZED TEST PROBLEM RESULTS

Problem		Golden Section	Cubic	Quadratic
1	TIME	1.23	1.19	1.26
	F	180.1	69.3	135.3
	G	17	84.2	17
2	TIME	.181	.181	.157
	F	69.1	29.7	49.2
	G	6	32.1	6
3	TIME	.123	.147	.1
	F	220.7	86.9	146.6
	G	15.7	88.4	15.7
4	TIME	2.42	2.67	2.28
	F	271.5	92.2	172.7
	G	565.2	644.5	567.2
5	TIME	5.18	5.15	5.10
	F	64.4	24.6	27.9
	G	2.8	24.0	3.0
6	TIME	4.59	4.29	4.44
	F	64.4	24.6	27.9
	G	2.8	24.0	3.0
7	TIME	3.94	4.02	3.98
	F	43.3	15.4	23.4
	G	99.7	112.9	100.4

TIME = time/single variable search
F = # function evaluations/single variable search
G = # gradient evaluations/single variable search

than Golden Section in 3 of the 7 problems with the same time for problem 2.

In comparing the number of function evaluations per single variable search, the quadratic interpolation method required fewer than the Golden Section on all seven problems. In most of the test problems, it required between thirty to forty per cent fewer function evaluations. The cubic interpolation method required fewer function evaluations than the other two methods because it used more information about the penalty function at each feasible step length to find the optimal step length. However, it required more gradient evaluations which increased the computation time making it about the same as the Golden Section and slightly slower than the quadratic interpolation.

With a larger sample of test problems more statistically sound comparisons could be made of the three different methods. Also samples of test problems that are similar in structure could be tested to show if one method worked better than the others on those type of problems.

It should be emphasized that the results in this thesis are obtained from single variable searches on a very special type of function — the unconstrained penalty function in the mixed interior-exterior penalty function algorithm. No attempt should be made to generalize these results to other nonlinear programming methods which also use single variable searches, but on a significantly different class of functions.

VI. CONCLUSIONS AND SUGGESTIONS FOR FURTHER STUDY

A. CONCLUSIONS

In looking at the computation time required for each single variable search, the quadratic interpolation method was faster than the other two methods on the test problems that were run. The quadratic interpolation method uses information about the penalty function at each feasible step length to reduce the interval of uncertainty whereas the Golden Section reduces the interval of uncertainty by a constant factor. Because of this difference, quadratic interpolation proved to be slightly more efficient than Golden Section. The only drawback in the quadratic interpolation method is the fact that bracketing the minimum can be quite hard to do. The cubic interpolation method reduced the interval of uncertainty in a more efficient manner than quadratic interpolation or Golden Section. However, having to make gradient evaluations to determine the directional derivative values proved to be very costly. The time per single variable search was about the same as the Golden Section search method.

If the user of the single variable search method has prior knowledge about the complexity of the function or gradient evaluations he may prefer cubic interpolation over Golden Section or quadratic interpolation, or vice versa, since the cubic interpolation requires more gradient evaluations and fewer function evaluations.

B. SUGGESTIONS FOR FURTHER STUDY

A combination of different methods made into a single variable search could be done with the three different methods that were programmed. This combined methods approach could use Golden Section until the minimum was bracketed, then use quadratic interpolation until the minimum was bracketed much closer and then finally use the cubic interpolation to home in on the local minimum of the search. The Lasdon, Fox and Ratner method uses three different stages that each find a feasible step length, with the final stage finding an optimal step length for the search. Another approximation of the penalty function like the Fletcher-McCann method could also be devised.

Implementation of the Lasdon, Fox and Ratner or Fletcher-McCann method into the SUMT program with runs of the test problems would enable a comparison to be made not only to the three methods tested in this paper but also between the Lasdon, Fox and Ratner method and Fletcher-McCann method.

Another possibility for study would be to fine tune the three different methods by adjusting the tolerances and termination conditions to see that if by allowing a less accurate determination of the minimum of the single variable search, the solution to the unconstrained problem could be found any faster.

If a larger sample of test problems including more and varied types of nonlinear programming problems could be tested using the three different methods, a better comparison

could be made. The results may either show one method to be superior over the other methods in all of the problems or show each method suited best to a specific type of problem enabling the user to choose the method which best applies to the situation at hand.

APPENDIX

GLOSSARY FOR GOLDEN SECTION OPT SUBROUTINE [Ref. 2]

DELX	The vector indicating the direction of move in one dimensional optimization. S
DELX0	The gradient vector of the penalty function ∇P.
DOTT	The inner product of the move vector and the negative gradient vector of the penalty function. $-S^T \nabla P$
I	Used as an index in DO loops.
ISW	A switch showing whether the motion on a given vector failed to decrease the penalty function and the negative was tried.
J	Used as an index in DO loops.
KSW	Switch showing whether less than 25 feasible points were found on the direction vector.
M	The number of inequality constraints
MN	Number of moves in search for a solution of a subproblem
N	The number of variables
NSATIS	Indicates whether constraints are satisfied.
NTCTR	The number of the point on which the program is working.
N401	The number of points generated in attempting to find a point along the search direction.
N404	The number of infeasible points found on the direction vector.
N405	Switch showing that a feasible point on the direction vector could not be found and the negative was tried.
PREV3	Penalty function value at lower step length P(X3).

PX1	Penalty function value for left interior steplength P(X2).
P0	Current value of the penalty function P(X)
P1	Penalty function value for upper steplength P(X1)
P31	Penalty function value at initial starting point
RJ	Vector of current values of the constraints
RJ1	Vector of previous value of the constraints
X	Vector of current value of the variables $X_0+\theta s$
XX	Temporary storage used for switching vector values
X1	X vector of upper step length $X_0+\theta_u s$
X2	X vector of left interior step length $X_0+\theta_a s$
X3	X vector of lower step length $X_0+\theta_L s$

```
      SUBROUTINE OPT
      IMPLICIT REAL*8(A-H,O-Z)
C
C              MARCH 1971
C
C OPT LOOKS FOR A MINIMUM ALONG THE SEARCH VECTOR USING THE
C GOLDEN SECTION SEARCH METHOD
      COMMON/SHARE/ X(100), DEL(100), A(100,100),N,M, MN,NP1
     1,NM1
      COMMON /VALUE/ F,G,PO,RSIGMA, RJ(200), RHO
      COMMON/CRST/ DELX(100), DELX0(100), RHOIN,RATIO, EPSI,
     1THETA0,
     2 RSIG1, G1,    X1(100), X2(100), X3(100), XR2(100),
     3XR1(100),PR1,
     4 PR2,P1, F1, RJ1(200), DOTT, PGRAD(100), DIAG(100),
     5 PREV3,ADELX, NTOTR, NUMINI, NPHASE, NSATIS
      DUMMY=DABS(DUMMY)
      KSW=1
      N405=1
      P31=PO
      ISW=1
      DOTT=0.
      DO 10 J=1,N
10    DOTT=DOTT+DELX(J)*DELX0(J)
      GO TO 40
20    DO 30 I=1,N
30    DELX(I)=-DELX(I)
40    CONTINUE
      N404=0
      MN=MN+1
C MN IS NOW NUMB. OF POINTS AFTER MIN ACHIEVED
      NTOTR=NTOTR+1
      DO 50 I=1,N
50    X2(I)=X(I)
      PX1=PO
      N401=0
60    N401=N401+1
      DO 70 I=1,N
70    X(I)=X2(I)+DELX(I)
      CALL EVALU
C 1 MEANS SATIS.OF CONSTRAINT NT.PREV. 2MEANS NOCHANGE 3
C MEANS VIOLATION
C IF POINT IS NOT FEASIBLE GIVE IT AN ARBRITRARILY HIGH VALU
      GO TO (540,90,80), NSATIS
80    PX2=10.E35
      PO=10.E35
      GO TO 100
90    CONTINUE
      PX2=PO
      IF (PX1-PX2) 100,100,150
100   IF (N401-2) 130,110,110
110   DO 120 I=1,N
120   X1(I)=X(I)
      P1=PX2
      GO TO 430
C ONLY ONE  POINT  SO FAR COMPUTED
130   DO 140 I=1,N
140   X3(I)=X2(I)
      PREV3=PX1
      GO TO 160
150   DO 160 I=1,N
      X3(I)=X2(I)
      X2(I)=X(I)
160   DELX(I)=1.6180339*DELX(I)
      PREV3=PX1
      PX1=PX2
      GO TO 60
C GOLDEN SECTION SEARCH METHOD.
C B VECTOR  GOES TO X1(I)
170   PO=1.E36
      N404=N404+1
180   DO 190 I=1,N
```

```
190     X1(I)=X(I)
        P1=PO
        DO 200 I=1,N
        X(I)=.3819660*(X1(I)-X3(I))+X3(I)
200     X2(I)=X(I)
        CALL EVALU
        GO TO (540,270,210), NSATIS
210     IF (N404.LT.30) GO TO 170
211     CONTINUE
C THERE IS NO REFERENCE TO 211, THE ABOVE STATEMENT IS A
C DUMMY STATEMENT
C--IT IS POSSIBLE NO FEASIBLE POINT EXIST, IF NOT TRY MOVING
C ON DELXO
C--   IF IT IS NOT POSSIBLE TO MOVE ON DELXO THEN WE MUST BE
C  AT A SOLUTION OF NLP PROBLEM.
        IF (N404.GT.100) GO TO 240
220     DO 230 I=1,N
        IF (ABS(ABS(X3(I)/X1(I))-1.).GT.1.E-7) GO TO 170
230     CONTINUE
240     GO TO (250,260), N405
250     N405=2
C--TRY TO MOVE ON GRADIENT.
        NTOTR=NTOTR-1
        MN=MN-1
        GO TO 20
260     WRITE (6,580)
        CALL TIMEC
        CALL OUTPUT (1)
        CALL REJECT
        STOP 22042
C
270     CONTINUE
        N404=0
        PX1=PO
        DO 280 I=1,N
280     X(I)=0.3819660*(X1(I)-X2(I))+X2(I)
        CALL EVALU
        GO TO (540,290,220), NSATIS
290     PX2=PO
        N401=1
300     N401=N401+1
        IF (N401-25) 340,310,310
310     KSW=2
        IF (N401-40) 320,460,460
320     DO 330 I=1,N
        IF (ABS(X2(I)/X(I)-1.0).GE.1.E-7) GO TO 340
330     CONTINUE
        GO TO 460
340     IF (ABS(PX1/PX2-1.).LE.1.E-7) GO TO 460
        IF (PX1-PX2) 350,460,400
C FROM LEFTORIGHT  X3(I)(PREV3)X2(I)(PX1)X(I)PX2 X1(I)P1
350     DO 360 I=1,N
360     X1(I)=X(I)
C THROW AWAY RIGHT PART
        P1=PX2
        DO 370 I=1,N
C POINTXP1 BECOMES XP2
370     X(I)=.3819660*(X1(I)-X3(I))+X3(I)
C TEMPORARILY IN  X STORAGE
        CALL EVALU
        GO TO (540,380,170), NSATIS
380     CONTINUE
        P>2=PX1
C SWITCH VECTORS TO PROPER POSITION
        PX1=PO
        DO 390 I=1,N
        XX=X2(I)
        X2(I)=X(I)
390     X(I)=XX
        GO TO 300
C LEFT SIDE  TOSSED  AWAY
C-- CHANGES FOR NONUNIMODAL FN
```

```
C-- GO TO THROW AWAY RIGHT IN THIS CASEINIT VAL LT FIB PT
400     IF (PREV3-PX2) 350,350,410
410     DO 420 I=1,N
        X3(I)=X2(I)
420     X2(I)=X(I)
        PREV3=PX1
        PX1=PX2
430     DO 440 I=1,N
440     X(I)=0.38196601*(X1(I)-X2(I))+X2(I)
        CALL EVALU
        GO TO (540,450,170), NSATIS
450     CONTINUE
        PX2=P0
        GO TO 300
C THE INTERIOR POINTS NOW GIVE EQUAL VALUE FOR P. COMPUTE MI
460     DO 470 I=1,N
        DELX0(I)=X(I)
        X(I)=(DELX0(I)+X2(I))*0.5
470     CONTINUE
        CALL EVALU
        GO TO (480,490), KSW
480     IF (ABS(P0/PX1-1.).GT.1.E-7) GO TO 520
490     GO TO (500,510), ISW
500     IF (P0.LT.P31) GO TO 510
        ISW=2
C IF P-FUNCTION DIDN,T  GO DOWN TRY NEG VECT.
        GO TO 20
510     RETURN
520     DO 530 I=1,N
530     X(I)=DELX0(I)
        GO TO 350
C ARE  WE NOW IN FEASIBILITY PHASE
540     DO 550 I=1,M
        IF (RJ(I)) 560,550,550
550     CONTINUE
        NSATIS=4
        RETURN
C---    PROBLEM HAS BECOME FEASIBLE
C ---    P - FUNCTION CHANGES IF A CONSTRAINT BECOMES FEASIBLE
560     MN=0
        DO 570 I=1,M
570     RJ1(I)=RJ(I)
        RETURN
C
580     FORMAT ( 80H  OPT CAN-T FIND A FEASIBLE POINT,THAT
       1GIVES A LOWER VALUE OF THE P-FUNCTION.          )
        END
```

GLOSSARY FOR CUBIC INTERPOLATION OPT SUBROUTINE

ADELX	Magnitude of the search direction vector
AL	Lower step length θ_L
BL	Upper step length θ_u
COTES	Cosine of the angle between the gradient and search direction vectors
D	Factor by which increment is multiplied
DELX	Search direction vector s
DELX0	Gradient vector of the penalty function $\nabla P(X)$
DLX1	Gradient vector of penalty function for lower step length $\nabla P(X3)$
DLX2	Gradient vector of penalty function for upper step length $\nabla P(X2)$
DOTS	Directional derivative of interpolated step length times ADELX P'(X)
DOTT	Directional derivative of initial starting point times ADELX P'(X1)
DOTTA	Directional derivative of lower step length times ADELX P'(X3)
DOTTB	Directional derivative of upper step length times ADELX P'(X2)
EPSO	Stopping criteria tolerance for cosine test
I	Index used in DO loops
J	Index used in DO loops
K	Index used in DO loops
KTER	Number of points evaluated in bracketing the minimum.
KTR	Number of cubic interpolations performed
M	Number of inequality constraints
MN	Number of moves in search for solution of a subproblem

N	Number of variables
NRED	Number of consecutive step length reductions
NSATIS	Indicates whether constraints are satisfied
NTCTR	Number of the point on which the program is working
N405	Switch showing that a feasible point on the direction vector could not be found and the negative was tried.
P0	Penalty function value at current xvector P(X)
PX1	Penalty function value at lower step length P(X3)
PX2	Penalty function value at upper step length P(X2)
Q	Coefficient used in computing interpolated step length
RJ	Vector of current values of the constraints
RJ1	Vector of previous values of the constraints
SMAG	Magnitude of the gradient of the penalty function
TAL	Current step length θ
TINC	Increment by which step's length increased or decreased
X	Current X vector
X1	Initial starting point X vector
X2	X vector of upper step length $X_0+\theta_u S$
X3	X vector of lower step length $X_0+\theta_L S$
Z	Coefficient used in computing interpolated step length

```
      SUBROUTINE OPT
C THIS OPT SUBROUTINE USES CUBIC INTERPOLATION TO FIND THE
C MINIMUM ALONG THE GIVEN SEARCH VECTOR
      IMPLICIT REAL*8(A-H,O-Z)
      COMMON/SHARE/ X(100), DEL(100), A(100,100),N,M, MN,NP1
     1,NM1
      COMMON /VALUE/ F,G,PO,RSIGMA, RJ(200), RHO
      COMMON/CRST/ DELX(100), DELXO(100), RHOIN,RATIO, EPSI,
     1THETAO,
     2 RSIG1, G1,    X1(100), X2(100), X3(100), XR2(100),
     3XR1(100),PR1,
     4 PR2,P1, F1, RJ1(200), DOTT, PGRAD(100), DIAG(100),
     5 PREV3,ADELX, NTCTR, NUMINI, NPHASE, NSATIS
      DIMENSION DLX1(100),DLX2(100)
      ABS(DUMMY)=DABS(DUMMY)
      SQRT(DUMMY)=DSQRT(DUMMY)
      EPSO=1.E-7
      DOTS=0.0
      N405=1
      GO TO 5
    3 DO 4 I=1,N
    4 DELX(I)=-DELX(I)
    5 CONTINUE
C INITIALIZATION
      MN=MN+1
      NTCTR=NTCTR+1
      DOTTA=0.0
      TAL=0.0
      AL=0.0
      D=2.0
      KTER=0
      NREC=0
      TINC=1.0
      SMAG=0.0
      DO 10 K=1,N
      SMAG=SMAG+DELX(K)**2
      DOTTA=DOTTA-DELX(K)*DELXO(K)
      X1(K)=X(K)
      DLX1(K)=DELXO(K)
   10 X3(K)=X(K)
      DOTT=-DOTTA
      FX1=PO
C INITIAL BRACKETING BEGINS
   25 TAL=TAL+TINC
   30 DO 35 K=1,N
   35 X(K)=X1(K)+DELX(K)*TAL
      KTER=KTER+1
      CALL EVALU
      GO TO (170,50,40), NSATIS
C REDUCE STEP SIZE
   40 D=0.5
      NREC=NREC+1
      TINC=(TAL-AL)*D
      TAL=TAL-TINC
      GO TO 30
C SECOND FEASIBLE POINT FOUND
   50 BL=TAL
      FX2=PO
      CALL GRAD(2)
      DOTTB=0.0
      DO 60 K=1,N
      X2(K)=X(K)
      DLX2(K)=DELXO(K)
   60 DOTTB=DOTTB-DELX(K)*DELXO(K)
C CHECK STOPPING CRITERIA OR MAYBE REVERSE SEARCH DIRECTION
      IF(ADELX.EQ.0.0) GO TO 91
      IF(SMAG.EQ.0.0) GO TO 91
      COTES= ABS(DOTTB)/(SQRT(SMAG)*ADELX)
      IF (COTES.LT.EPSO) GO TO 91
      IF(KTER.GT.100) GO TO 280
      IF(NREC.GT.20) GO TO 280
      DO 75 K=1,N
```

```
      IF(ABS(ABS(X1(K)/X2(K))-1.).GT.1.E-7) GO TO 76
   75 CONTINUE
      GO TO 280
   76 IF(ABS(ABS(PX2/PX1)-1.).LE.1.E-7) GO TO 91
C IF DERIVATIVE IS POSITIVE THEN THE MINIMUM IS BRACKETED
C AND CUBIC INTERPOLATION CAN BE PERFORMED, IF NOT MAKE THIS
C POINT THE LOWER STEP LENGTH AND STEP OUT FARTHER
      IF(DOTTB.GT.0.0) GO TO 100
      NREC=0
      TINC=TINC*D
      DO 80 K=1,N
      DLX1(K)=DLX2(K)
   80 X3(K)=X2(K)
      PX1=PX2
      AL=BL
      DOTTA=DOTTB
      GO TO 25
C OF THE TWO FEASIBLE POINTS RETURN WITH THE SMALLEST
   91 IF(PX2.LE.PX1)  GO TO 220
      PO=PX1
      DO 92 K=1,N
      X(K)=X3(K)
      DELXO(K)=DLX1(K)
   92 CONTINUE
      GO TO 220
C CUBIC INTERPOLATION FOLLOWS ********************************
  100 KTR=0
  105 CONTINUE
      Z=3.*((PX1-PX2)/(BL-AL))+DOTTA+DOTTB
      Q=SQRT(Z**2-DOTTA*DOTTB)
      TAL=BL-(BL-AL)*(DOTTB+Q-Z)/(DOTTB-DOTTA+2.*Q)
      DO 130 K=1,N
  130 X(K)=X1(K)+DELX(K)*TAL
      CALL EVALU
      GO TO (170,135,40), NSATIS
  135 CALL GRAD(2)
      DOTS=0.0
      DO 140 K=1,N
      DOTS=DOTS-DELX(K)*DELXO(K)
  140 CONTINUE
C CHECK STOPPING CRITERION OR IF SEARCH DIRECTION SHOULD BE
C REVERSED
      IF(ADELX.EQ.0.0) GO TO 471
      IF(SMAG.EQ.0.0) GO TO 471
      COTES=ABS(DOTS)/(SQRT(SMAG)*ADELX)
      IF (COTES.LT.EPSO) GO TO 471
      IF(ABS(ABS(PX2/PO )-1.).LE.1.E-7) GO TO 471
      KTR=KTR+1
      IF(KTR.GT.10) GO TO 471
      DO 143 K=1,N
      IF(ABS(ABS(X2(K)/X(K) )-1.).GT.1.E-7) GO TO 163
  143 CONTINUE
      GO TO 220
  163 IF(DOTS.GT.0.0) GO TO 150
C THROW AWAY LEFT SIDE
      AL=TAL
      PX1=PO
      DOTTA=DOTS
      DO 145 K=1,N
      DLX1(K)=DELXO(K)
  145 X3(K)=X(K)
      GO TO 105
C THROW AWAY RIGHT SIDE
  150 BL=TAL
      PX2=PO
      DOTTB=DOTS
      DO 160 K=1,N
      DLX2(K)=DELXO(K)
  160 X2(K)=X(K)
      GO TO 105
  170 DO 180 K=1,M
      IF(RJ(K)) 190,190,180
```

```
  180 CONTINUE
      NSATIS=4
      RETURN
  190 MM=0
      DO 200 K=1,M
  200 RJ1(K)=RJ(K)
      RETURN
C RETURN WITH SMALLEST VALUE
  471 IF(PO.LE.PX1) GO TO 474
      IF(PX1.LE.PX2) GO TO 475
  472 PO=PX2
      DO 473 K=1,N
      X(K)=X2(K)
      DELXO(K)=DLX2(K)
  473 CONTINUE
      GO TO 220
  474 IF(PO.LE.PX2) GO TO 220
      GO TO 472
  475 PO=PX1
      DO 476 K=1,N
      X(K)=X3(K)
      DELXO(K)=DLX1(K)
  476 CONTINUE
  220 CONTINUE
  225 CONTINUE
      RETURN
C REVERSE SEARCH DIRECTIONS
  280 GO TO (290,300),N405
  290 N405=2
      NTCTR=NTCTR-1
      MM=MM-1
      GO TO 3
  300 WRITE(6,580)
      CALL TIMEC
      CALL OUTPUT(1)
      CALL REJECT
  580 FORMAT( 80H  OPT CAN-T FIND A FEASIBLE POINT,THAT
     1GIVES A LOWER VALUE OF THE P-FUNCTION.          )
      STOP 22042
      END
```

GLOSSARY FOR QUADRATIC INTERPOLATION OPT SUBROUTINE

AL	Interpolated step length
D	Factor by which increment is multiplied
DELX	Vector indicating the direction of move in one dimensional optimization S
DELXO	Gradient vector of the penalty function ∇P
DOTT	Directional derivative of initial starting point times search direction magnitude P'(X1)
I	Index used in DO loops
J	Index used in DO loops
K	Index used in DO loops
KTER	Number of points evaluated in bracketing the minimum
KTR	Number of quadratic interpolations performed
M	Number of inequality constraints
MN	Number of moves in search for solution of a subproblem
N	Number of variables
NFIRS	Indicates if more than two feasible step lengths found
NRED	Number of consecutive step length reductions
NSATIS	Indicates whether constraints are satisfied
NTCTR	Number of the point on which the program is working
N405	Switch showing that a feasible point on the direction vector could not be found and the negative was tried
PO	Current penalty function value P(X)
PX1	Penalty function value of lower step length P(X3)

PX2	Penalty function value of upper step length
RJ	Vector of previous values of the constraints
TAL	Current step length θ
TAL1	Lower step length θ_L
TAL2	Upper step length θ_u
TINC	Increment by which step length increased or decreased
X	Current X vector $X_0 + \theta s$
XX	Temporary storage used for switching values
X1	Initial starting point vector X_0
X2	X vector at upper step length $X_0 + \theta_u s$
X3	X vector of lower step length $X_0 + \theta_L s$

```
      SUBROUTINE OPT
C THIS OPT SUBROUTINE FINDS THE MINIMUM ALONG A GIVEN SEARCH
C VECTOR USING QUADRATIC INTERPOLATION
      IMPLICIT REAL*8(A-H,O-Z)
      COMMON/SHARE/ X(100), DEL(100), A(100,100),N,M, MN,NP1
     1,NM1
      COMMON /VALUE/ F,G,PO,RSIGMA, RJ(200), RHO
      COMMON/CRST/ DELX(100), DELXO(100), RHOIN,RATIO, EPSI,
     1THETAO,
     2 RSIG1, G1,    X1(100), X2(100), X3(100), XR2(100),
     3XR1(100),PR1,
     4  PP2,P1, F1, RJ1(200), DOTT, PGRAD(100), DIAG(100),
     5 PREV3,ADELX, NTCTR, NUMINI, NPHASE, NSATIS
      AES(DUMMY)=DABS(DUMMY)
      N405=1
      DOTT=0.
      DO 2 K=1,N
2     DOTT=DOTT+DELX(K)*DELXO(K)
      GO TO 5
3     DO 4 I=1,N
4     DELX(I)=-DELX(I)
5     CONTINUE
C INITIALIZATION
      MN=MN+1
      NTCTR=NTCTR+1
      TAL=0.0
      TAL1=0.0
      D=2.
      KTER=0
      NREC=0
      TINC=1.0
      DO 7 K=1,N
      X1(K)=X(K)
      X3(K)=X(K)
7     PX1=PO
      NFIRS=0
C START BRACKETING PROCEDURE
      TAL=TAL+TINC
10    DO 13 K=1,N
13    X(K)=X1(K)+DELX(K)*TAL
      KTER=KTER+1
      CALL EVALU
      GO TO (170,20,20), NSATIS
C REDUCE STEP SIZE
20    D=0.5
      IF(NREC.GT.20) GO TO 280
      NREC=NREC+1
      TINC=(TAL-TAL1)*D
      TAL=TAL-TINC
      GO TO 10
C HAVE 2 OR MORE FEASIBLE POINTS BEEN FOUND
30    IF(NFIRS.EQ.1) GO TO 40
      NFIRS=1
31    DO 32 K=1,N
32    X2(K)=X(K)
      PX2=PO
      TAL2=TAL
      IF(PX2.GE.PX1) GO TO 20
C INCREASE STEP SIZE
35    TINC=TINC*D
      TAL=TAL+TINC
      NREC=0
      GO TO 10
40    CONTINUE
C RECORDER POINTS
      IF(TAL.LT.TAL2) GO TO 54
      IF(PO.LT.PX2) GO TO 50
      XX=TAL
      TAL=TAL2
      TAL2=XX
      XX=PO
      PO=PX2
```

```
      PX2=XX
      DO 45 K=1,N
      XX=X(K)
      X(K)=X2(K)
45    X2(K)=XX
      GO TO 54
46    TAL1=TAL
      PX1=PO
      DO 47 K=1,N
47    X3(K)=X(K)
      TAL=TAL2
      GO TO 35
50    TAL1=TAL2
      PX1=PX2
      DO 51 K=1,N
      X3(K)=X2(K)
51    X2(K)=X(K)
      TAL2=TAL
      PX2=PO
C CHECK STOPPING CRITERION OR IF SEARCH DIRECTION SHOULD BE
C REVERSED
      IF(ABS(ABS(PO/PX1)-1.).LT.1.E-7) GO TO 220
      DO 52 K=1,N
      IF(ABS(ABS(X1(K)/X(K))-1.).GT.1.E-7) GO TO 53
52     CONTINUE
      GO TO 280
53    CONTINUE
      GO TO 35
C CHECK STOPPING CRITERION OR IF SEARCH DIRECTION SHOULD BE
C REVERSED
54    IF(ABS(ABS(PO/PX1)-1.).LT.1.E-7) GO TO 220
      DO 55 K=1,N
      IF(ABS(ABS(X1(K)/X(K))-1.).GT.1.E-7) GO TO 56
55    CONTINUE
      GO TO 280
56    CONTINUE
      IF(KTER.GE.100) GO TO 280
      IF(PO.GT.PX2) GO TO 46
      IF(PO.GE.PX1) GO TO 31
      DO 57 K=1,N
      IF(ABS(ABS(X2(K)/X(K))-1.).GE.1.E-7) GO TO 60
57    CONTINUE
      GO TO 220
C QUADRATIC INTERPOLATION BEGINS**************************
60    KTR=0
61    KTR=KTR+1
      AL=0.5*((TAL**2-TAL2**2)*PX1+(TAL2**2-TAL1**2)*PO+
     1(TAL1**2-TAL**2
     2)*PX2)/((TAL-TAL2)*PX1+(TAL2-TAL1)*PO+(TAL1-TAL)*PX2)
      IF(TAL.GT.AL) GO TO 75
C THROW AWAY LEFT SIDE
      TAL1=TAL
      PX1=PO
      DO 71 K=1,N
71    X3(K)=X(K)
      GO TO 78
C THROW AWAY RIGHT SIDE
75    TAL2=TAL
      PX2=PO
      DO 77 K=1,N
77    X2(K)=X(K)
78    TAL=AL
      DO 79 K=1,N
79    X(K)=X1(K)+DELX(K)*TAL
      CALL EVALU
      GO TO (170,80,85), NSATIS
80    DO 82 K=1,N
C CHECK STOPPING CRITERION OR IF SEARCH DIRECTION SHOULD BE
C REVERSED
      IF(ABS(ABS(X2(K)/X(K))-1.).GE.1.E-7) GO TO 83
82    CONTINUE
      GO TO 220
```

```
83      IF(ABS(ABS(PO/PX1)-1.).LT.1.E-7) GO TO 220
        IF(KTR.GT.20) GO TO 220
        GO TO 61
85      NREC=0
        KTER=0
        GO TO 20
  170   DO 180 K=1,M
        IF(RJ(K)) 190,190,180
  180   CONTINUE
        NSATIS=4
        RETURN
  190   MN=0
        DO 200 K=1,M
  200   RJ1(K)=RJ(K)
        RETURN
  220   CONTINUE
C RETURN WITH SMALLEST VALUE
        IF(PX1.GE.PO) GO TO 240
        PO=PX1
        DO 230 K=1,N
230     X(K)=X3(K)
240     CONTINUE
        RETURN
C REVERSE SEARCH DIRECTIONS
  280   GO TO (290,300),N405
  290   N405=2
        NTCTR=NTCTR-1
        MN=MN-1
        GO TO 3
  300   WRITE(6,580)
        CALL TIMEC
        CALL OUTPUT(1)
        CALL REJECT
  580   FORMAT( 80H  OPT CAN-T FIND A FEASIBLE POINT,THAT
     1GIVES A LOWER VALUE OF THE P-FUNCTION.          )
        STOP 22042
        END
```

LIST OF REFERENCES

1. Mylander, W.C., Holmes, R.L., and McCormick, G.P., SUMT-Version 4, FORTRAN IV Nonlinear Programming Compute Code, Research, Analysis Corporation, McLean, Va., March 1971.

2. Mylander, W.C., Holmes, R.L., and McCormick, G.P., A Guide to SUMT-Version 4: The Computer Program Implementing the Sequential Unconstrained Minimization Technique for Nonlinear Programming, Research Analysis Corporation, McLean, Va., February 1971.

3. Fiacco, A.V. and McCormick, G.P., Nonlinear Programming Sequential Unconstrained Minimization Techniques, John Wiley and Sons, Inc., 1968.

4. Lasdon, L.S., Fox, R.L., and Ratner, M.W., "An Efficient One-Dimensional Search Procedure for Barrier Functions", Mathematical Programming, Vol. 4, No. 3, p. 279-296, 1973.

5. Fletcher, R., Optimization, p. 210-213, Academic Press, 1969.

6. Luenberger, D.G., Introduction to Linear and Nonlinear Programming, p. 134-137, Addison-Wesley, 1965.

7. Waterman, R.J., An Evaluation and Comparison of Three Nonlinear Programming Codes, Master's Thesis, Naval Postgraduate School, Monterey, March 1976.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Documentation Center Cameron Station Alexandria, Virginia 22314	2
2.	Library, Code 0212 Naval Postgraduate School Monterey, California 93940	2
3.	Department Chairman, Code 55 Department of Operations Research Naval Postgraduate School Monterey, California 93940	1
4.	Professor J.K. Hartman, Code 55Hh Department of Operations Research Naval Postgraduate School Monterey, California 93940	3
5.	Professor G.H. Bradley, Code 55Bz Department of Operations Research Naval Postgraduate School Monterey, California 93940	1
6.	Professor W.C. Mylander, Code Department of Operations Analysis U.S. Naval Academy Annapolis, Maryland 21401	1
7.	Associate Professor G.G. Brown, Code 55Zr Department of Operations Research Naval Postgraduate School Monterey, California 93940	1
8.	Ensign D.B. Wick, USN 505 Road 38 Pasco, Washington 99301	2